Fatine Maataoui
Cinta Gonzalez

Mutty

Open Questions and its experiments

i) Make tests with different `Sleep` and `Work` parameters
to analyze how this lock implementation responds to different contention degrees.

*****Foto de la taula amb diferents paràmetres insertada a l'annex*****

ii) Split the `muty` module and make the needed adaptations to enable each worker-lock pair to run in different machines (that is, john and l1 should run in a machine, ringo and l2 in another, and so on). Remember
how names registered in remote nodes are referred and how Erlang runtime should be started to run distributed programs.

Si s'executen els workers a diferents nodes es pot veure com el temps mig per aconseguir el lock augmenta lleugerament, a causa de l'overhead, que comporta la comunicació entre els diferents nodes.

Lock 1: What is the behavior of the lock when you increase the risk of a conflict?
Si s'incrementa el lock de risk també augmenten els deadlocks, el qual provoca que augmente el nombre de withdrawals.

Repeat the previous tests to compare the behavior of
this lock with respect to the previous one.

Lock 2:
i) Justify how your code guarantees that only one process is in the critical section at any time.

Els locks están inactius (open mode) donen un OK a tothom.
Es vol accedir a la critical path , el lock es posa a wait fins que rebi un ok dels altres nodes.
Cada cop que s'allibera la zona crítica, el lock queda notificat, així es sap quan un node pot entrar-hi.

ii) What is the main drawback of `lock2` implementation?

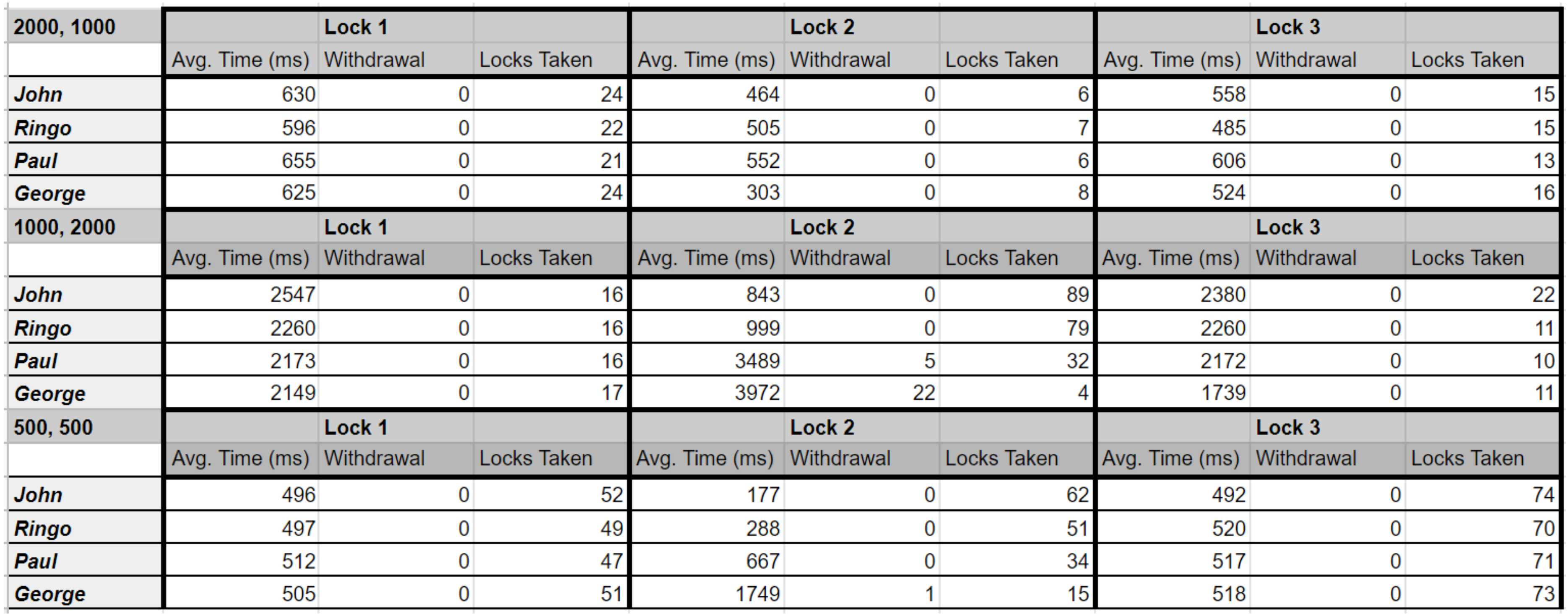
Els nodes amb prioritat molt baixa tindran com a conseqüència molts withdrawals.

*Repeat the previous tests to compare this version with
the former ones.*

Lock 3: Note that the workers are not involved in the Lamport clock. According to this, would it be possible that a worker is given access to a critical section prior to another worker that issued a request to its lock instance before (assuming real-time order)?

Sí que podria arribar a passar, no obstant, el temps seria d'una diferència mínima (microsegons, ms).
Això passa perquè no es contempla el temps lògic als workers, només es fa a les instàncies del lock.

Annex



2000, 1000		**Lock 1**			**Lock 2**			**Lock 3**	
	Avg. Time (ms)	Withdrawal	Locks Taken	Avg. Time (ms)	Withdrawal	Locks Taken	Avg. Time (ms)	Withdrawal	Locks Taken
John	630	0	24	464	0	6	558	0	15
Ringo	596	0	22	505	0	7	485	0	15
Paul	655	0	21	552	0	6	606	0	13
George	625	0	24	303	0	8	524	0	16
1000, 2000		**Lock 1**			**Lock 2**			**Lock 3**	
	Avg. Time (ms)	Withdrawal	Locks Taken	Avg. Time (ms)	Withdrawal	Locks Taken	Avg. Time (ms)	Withdrawal	Locks Taken
John	2547	0	16	843	0	89	2380	0	22
Ringo	2260	0	16	999	0	79	2260	0	11
Paul	2173	0	16	3489	5	32	2172	0	10
George	2149	0	17	3972	22	4	1739	0	11
500, 500		**Lock 1**			**Lock 2**			**Lock 3**	
	Avg. Time (ms)	Withdrawal	Locks Taken	Avg. Time (ms)	Withdrawal	Locks Taken	Avg. Time (ms)	Withdrawal	Locks Taken
John	496	0	52	177	0	62	492	0	74
Ringo	497	0	49	288	0	51	520	0	70
Paul	512	0	47	667	0	34	517	0	71
George	505	0	51	1749	1	15	518	0	73